Claims

[c1] *1*. A system allowing mobile device to run multiple operating systems and preserve the state and data of the original operating system, comprising the steps of:
- Preparing a guest OS image;
- Packaging the image into a special host OS application or file;
- Starting a special boot loader for the host OS to read, unpack, load and start the guest OS image from the special host OS application;
- Preserving the current state and data of the host OS and starting the guest OS;
- Exiting from the guest OS, running exiting code to restore the system state and data to the original host OS and then return back to the host OS.

[c2] 2. The system of claim 1, wherein the mobile device is a PDA or Win CE device, cellular phone or other mobile devices.

[c3] 3. The system of claim 1, wherein the original host OS cannot share its memory with any other OS concurrently.

[c4] 4. The system of claim 1, wherein the original host OS

can only start applications packed with some special formats.

[c5] 5. The system of claim 1, further comprising a step of preserving the state and data of the guest OS when user wants to temporarily switch away from the guest OS.

[c6] 6. The system of claim 1, wherein the boot loader is either a standalone host OS application or built into the guest OS image wrapper application.

[c7] 7. The system of claim 1, wherein the guest OS image is compressed.

[c8] 8. The system of claim 1, wherein the guest OS will use up all memories available.

[c9] 9. The system of claim 1, further comprising a step of saving the state and data of the guest OS, so users will not lose its work in the guest OS and can return back after switching to original host OS.

[c10] 10. The system of claim 1, further comprising starting additional guest OS from either host OS or within the guest OS using the same procedure.

[c11] 11. The system of claim 1, wherein the guest OS image is stored in a memory card as a regular file while host OS can access regular files in memory card.

[c12] 12. The system of claim 1, wherein the guest OS image is stored in a memory card and can be started immediately when the card is inserted into the device.

[c13] 13. The system of claim 1, wherein the boot loader only preserves the modified or used memory of host OS and do not care about free or unused memory blocks in the system.

[c14] 14. The system of claim 1, wherein IO states or registers of peripheries are saved in memories and preserved the same as other memories and then after memories are restored, restore IO states or registers from the memories.

[c15] 15. The system of claim 1, wherein the host OS or guest OS can be Palm OS, Windows CE, Symbian, Embedded Linux or other mobile operating systems.

[c16] *16.* A method of packaging guest OS image inside a host OS application that allows in-place execution of the guest OS code to reduce memory usage where the host OS can only recognized some special file formats:
When compiler supports, compiling guest application code to skip those areas required by host OS file format such as record headers as if they are unreachable blocks; or

Use dynamic linking and jump tables for each code segment embedded in the wrapper host application and then filling out the jump tables at run time after loading.

[c17] 17. The method of claim 16, wherein the host OS file format is a database file with multiple records or Palm PDB file format.

[c18] 18. The method of claim 16, wherein the guest OS code requires sequential arrangement in memory.

[c19] 19. The method of claim 16, further comprising a step of splitting guest kernel image and other related files into different pieces and wrapped into multiple host OS files.

[c20] *20*. A method to preserve the state and data of the host OS in a mobile device where the boot loader manipulates the memory protection bits for pages of the memory if both OS use the MMU (Memory Management Unit) of the mobile device which supports memory protection; changing attributes of all memories used in the original OS (data and code) from read/write to read-only (not write-able) or even not accessible so that those memories of the host OS can no longer be modified after switching to guest OS.

[c21] 21. The method of claim 20, wherein the memory protection works on physical memory space.

[c22] 22. The method of claim 20, wherein the memory protection works on virtual memory space and both OS share the same virtual memory.

[c23] *23*. A method to preserve the state and data of the host OS in a mobile device to move/copy data of the host OS to the top of the system memories and faked a reduced memory environment to the guest OS. When the guest OS starts, it will think that it is running on a reduced memory device and will never touch memories above the told highest available memory that are used to backup the host OS data, thus the data and state of the host OS is preserved.

[c24] 24. The method of claim 23, wherein the boot loader writes to a special register to fake the available memories to guest OS.

[c25] 25. The method of claim 23, wherein the guest OS changed its memory detection module to assume the faked maximum memory size.

[c26] 26. The method of claim 23, wherein the guest OS can have its own memory management module or virtual memory space as long as it use only memories it detected on booting.

[c27] 27. The method of claim 23, wherein the memories of mobile devices are divided into several banks or memory addresses are not continuous and the booting system fake that one memory bank is no longer available.

[c28] *28*. A method to use a special memory device driver loaded during the initialization of the guest OS to claim memories of the host OS. The special memory device driver can claim those memories currently used by the host OS (including data, code and runtime) and keep its content from being modified by other programs of the guest OS until guest OS exit.

[c29] 29. The method of claim 28, wherein the special memory device driver mark the memories as inaccessible to the other part of the guest OS.

[c30] 30. The method of claim 28, wherein the special memory device driver is a kernel mode driver in guest OS that prevent the memory from being used by both kernel and user space processes.

[c31] 31. The method of claim 28, wherein if both OS need to use a same memory location, it may temporarily move the memory of the host OS to free areas and move back upon termination of the guest OS.

[c32] *32*. A method to preserve the state and data of the host

OS in a mobile device by backing up the current state of the host OS to an external memory cards. The process is: boot loader first save the whole OS memory image into external memory device and an exit code restore the host OS from the memory image from the external card upon retuning from the guest OS.

[c33] 33. The method of claim 32, wherein the memory cards are Compact Flash (CF) memory card or Secure Digital (SD) memory card or MMC card.

[c34] 34. The method of claim 32, wherein the whole memory images can be saved to memory cards as a backup point and being restored later.

[c35] 35. The method of claim 32, wherein the saved memory images can transferred to another mobile device, emulator or networks for mirroring purpose.